1.Write a blog on difference between HTTP1.1 vs HTTP2

**Transmission Model**

The first difference between HTTP1.1 and http2 is the transport [mode](https://developpaper.com/tag/mode/)l.

In http1.0, the resource request is a request and a response, and almost all TCP are short links (unless the keepalive feature is actively turned on). For a [page](https://developpaper.com/tag/page/) request, it usually contains many resources. If you need to render a page with multiple resources, just one response and one answer, you need to repeatedly build multiple TCP connections, which increases the resource overhead. Therefore, in HTTP1.1, keep alive feature is opened by default, and requests from multiple resources can take the same TCP, which reduces the overhead of chain building and chain breaking. This approach is called pipeline. The problem with pipeline is that although TCP is used, the requests for resources are serial. If the resource requests in the front row are blocked, the subsequent resource transmission will be affected. This is called hol (head of line) blocking. If we adopt the strategy of building multiple TCP links in parallel to solve this problem, both the client and the server will face higher overhead. Especially for the server, when the number of concurrent connections is limited, the number of concurrent client services will be greatly reduced.

Http2 uses the bottom stream technology, which has no influence on the upper layer of HTTP semantics, but in the data stream transmission, plain text is no longer used. When a request contains requests for multiple resources, different resources are mapped to different binary streams, each stream has a unique ID, and the interdependence between different stream resources is described by the parent field. At the same time, each flow can also specify the priority, and the higher the priority number is, the higher the response priority will be. For the data of a resource, the data is further divided into smaller units, which is called frame. In a flow channel (the flow channel is based on TCP protocol), different ID streams can be transmitted at the same time to realize the parallel transmission of multiple resources, and the sequence of resource transmission can be flexibly customized by defining the priority.

**Flow control**

The second difference is the change of implementation mode on flowcontrol.

For both the client and the server, there is a buffer to cache incomplete information. When the information in the cache is complete, it will be processed together. Considering that buffer overflow may occur when large resources need to be transmitted, sliding window is adopted in TCP protocol to avoid buffer overflow. The receiver receives data and sends ack. In ACK, it can report the remaining free space in the local buffer. The sender can adjust the sending rate according to the reported free space. Once the free space is 0, the sender stops sending. When the receiver has finished processing the data in the buffer, it requests the sender to continue sending. The flow control of HTTP1.1 depends on the sliding window of TCP.

Since http2.0 puts forward the transmission mode of parallel stream, its flow control is also agreed by stream. This Convention is put into the application layer, that is, the application can dynamically negotiate the cache size based on its own buffer request. At the same time, in the end-to-end transmission process, the intermediate nodes can also identify the flow control parameters, so as to achieve better flow control.

**Predicting Resource Requests**

The third difference is the strategy of resource pre request.

In an HTTP request of a page, in order to render a page, you need to request CSS as well as HTML page file. In HTTP1.1, if it is predicted that some resources will be sent in the subsequent request, it can respond to the client in the first request. This is called resource inlining. But there are some problems in this way. For example, it may repeatedly respond to the existing resources of the client. When the resources being inline are very large, the waste of network resources is also very large. Therefore, in practice, this way is rarely used.

Http2 proposes a method called server push. If it is predicted that some resources may be subsequently requested, a push message is first pushed to the client\_ Promise frame, which describes the metadata of the content to be pushed. If the client does not need some resources, it can respond to an rst\_ Stream to cancel some resources. This avoids the waste of resources. At the same time, the client can also sendSETTINGSFrame to change the behavior of server push.

**Compression**

The last difference is compression. HTTP1.1 only compresses the message body, not the HTTP header, because the header is usually very small. However, when the number of requests is large, the overhead of the header on the network bandwidth will also increase. Http2 defines hpack to compress the header, especially when there is only partial difference between two requests or replies, only the difference is transmitted.